



US 20070101433A1

(19) **United States**

(12) **Patent Application Publication**
Louch et al.

(10) **Pub. No.: US 2007/0101433 A1**

(43) **Pub. Date: May 3, 2007**

(54) **WIDGET SECURITY**

Related U.S. Application Data

(76) Inventors: **John O. Louch**, San Luis Obispo, CA (US); **Scott Forstall**, Mountain View, CA (US); **Eric Steven Peyton**, Lisle, IL (US)

(60) Provisional application No. 60/730,956, filed on Oct. 27, 2005. Provisional application No. 60/734,016, filed on Nov. 4, 2005. Provisional application No. 60/737,899, filed on Nov. 18, 2005.

Publication Classification

Correspondence Address:

FISH & RICHARDSON P.C.

PO BOX 1022

MINNEAPOLIS, MN 55440-1022 (US)

(51) **Int. Cl.**
G06F 11/00 (2006.01)

(52) **U.S. Cl.** **726/25**

(57) **ABSTRACT**

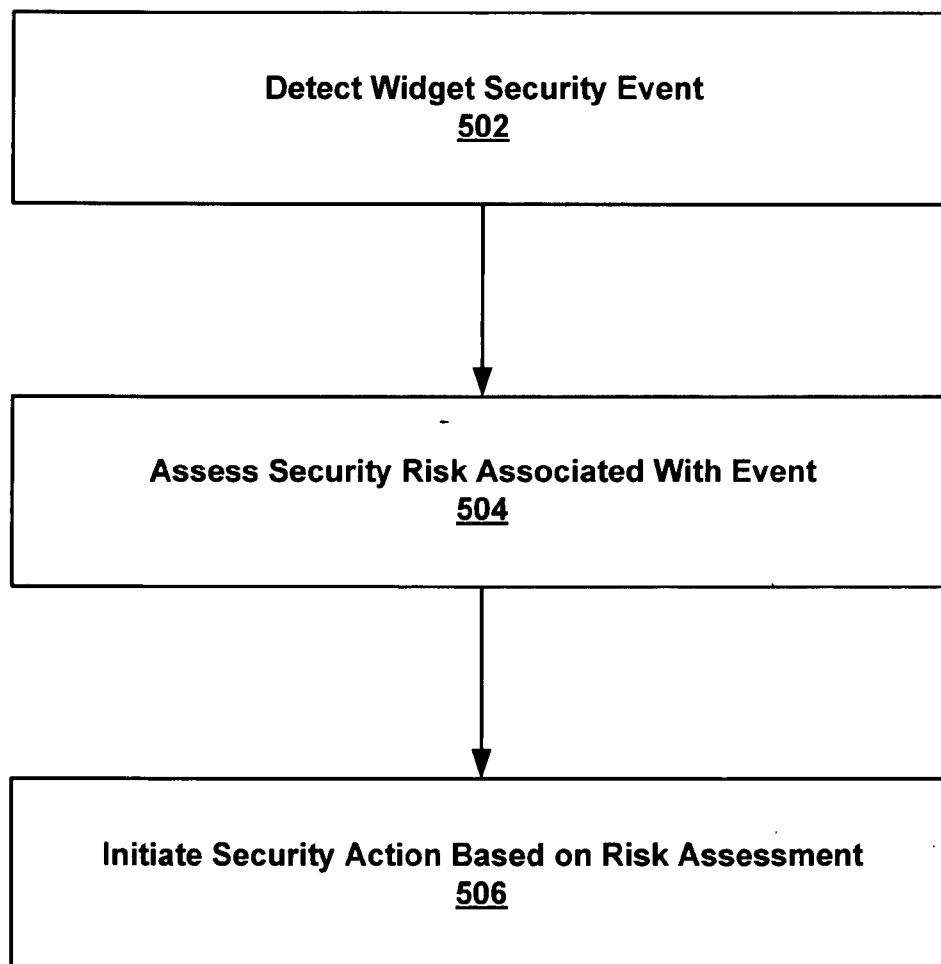
A widget security system, method and computer-readable medium detects a security event associated with a widget, assesses the risk associated with the security event, and initiates a security action based on the assessed risk.

(21) Appl. No.: **11/432,295**

(22) Filed: **May 10, 2006**

Widget Security Process

500



Hardware Architecture
100

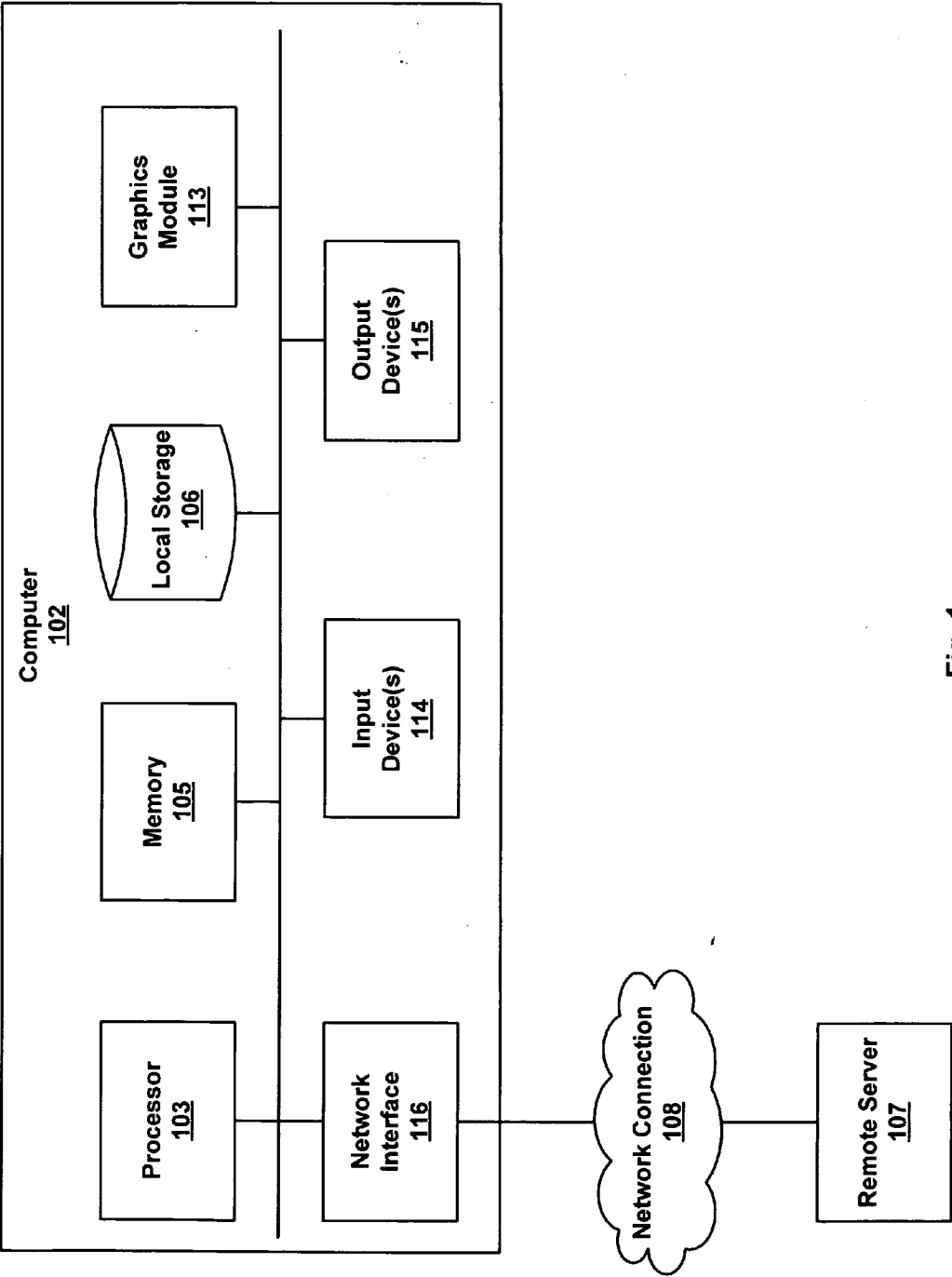


Fig. 1

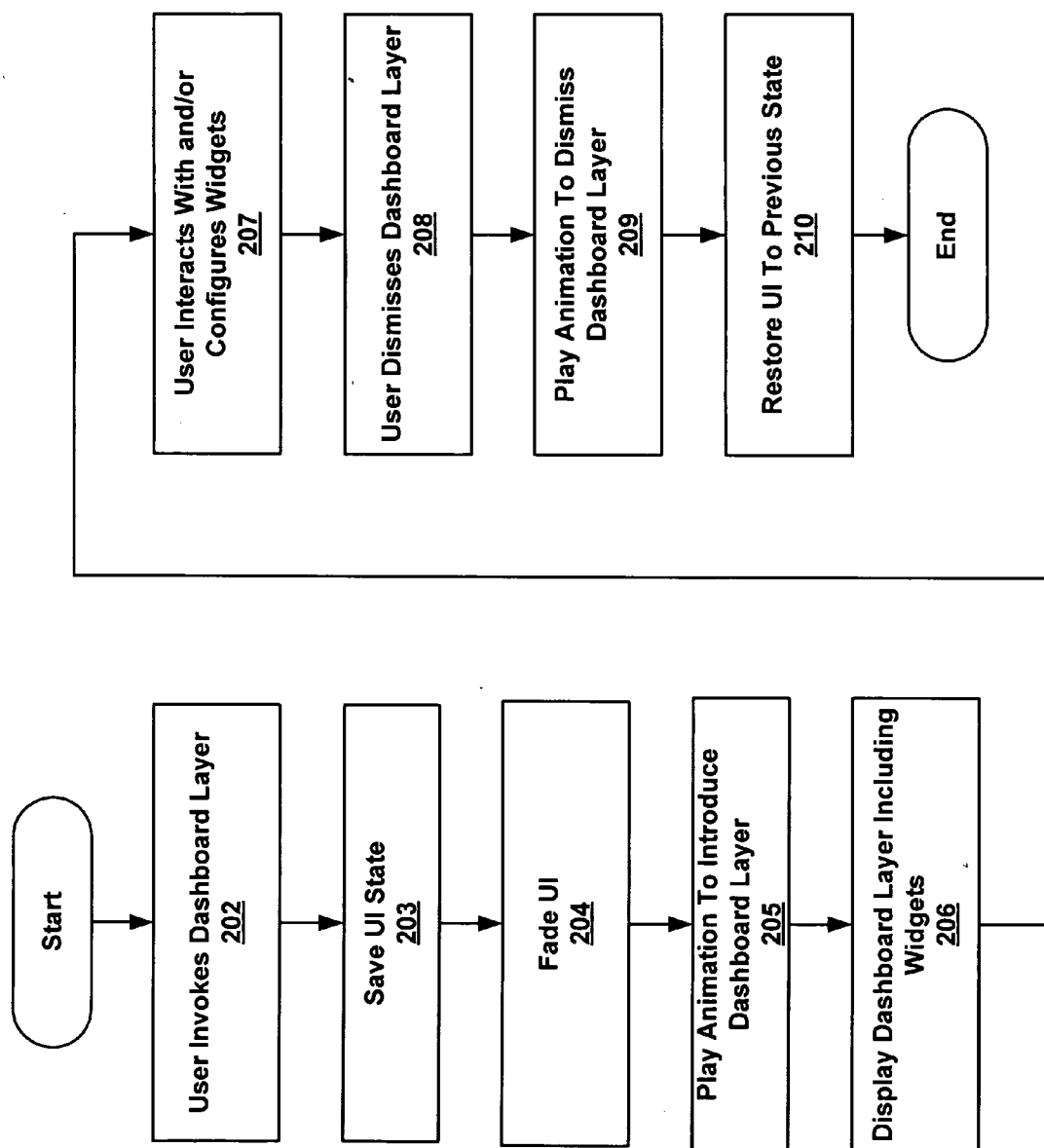


Fig. 2

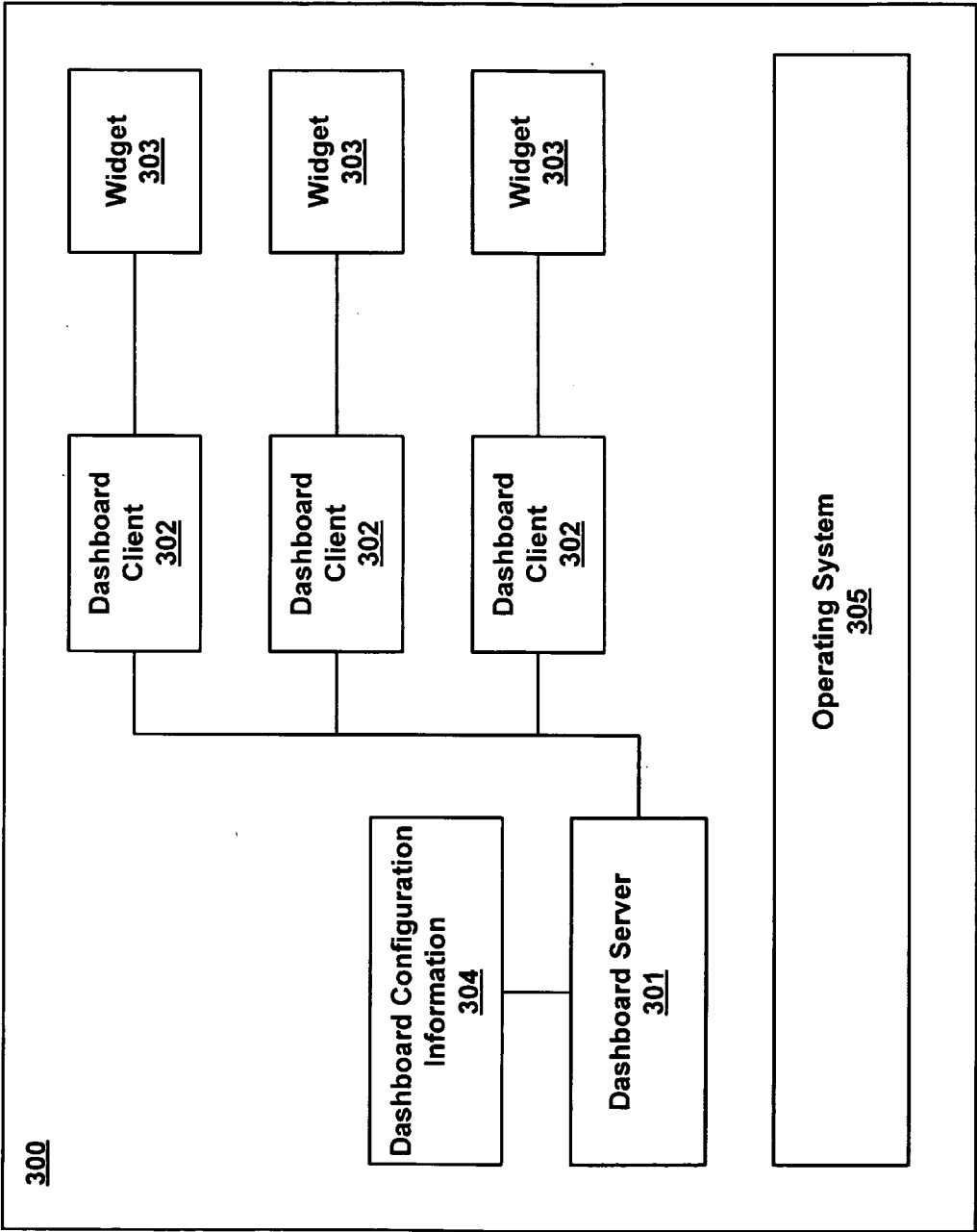


Fig. 3

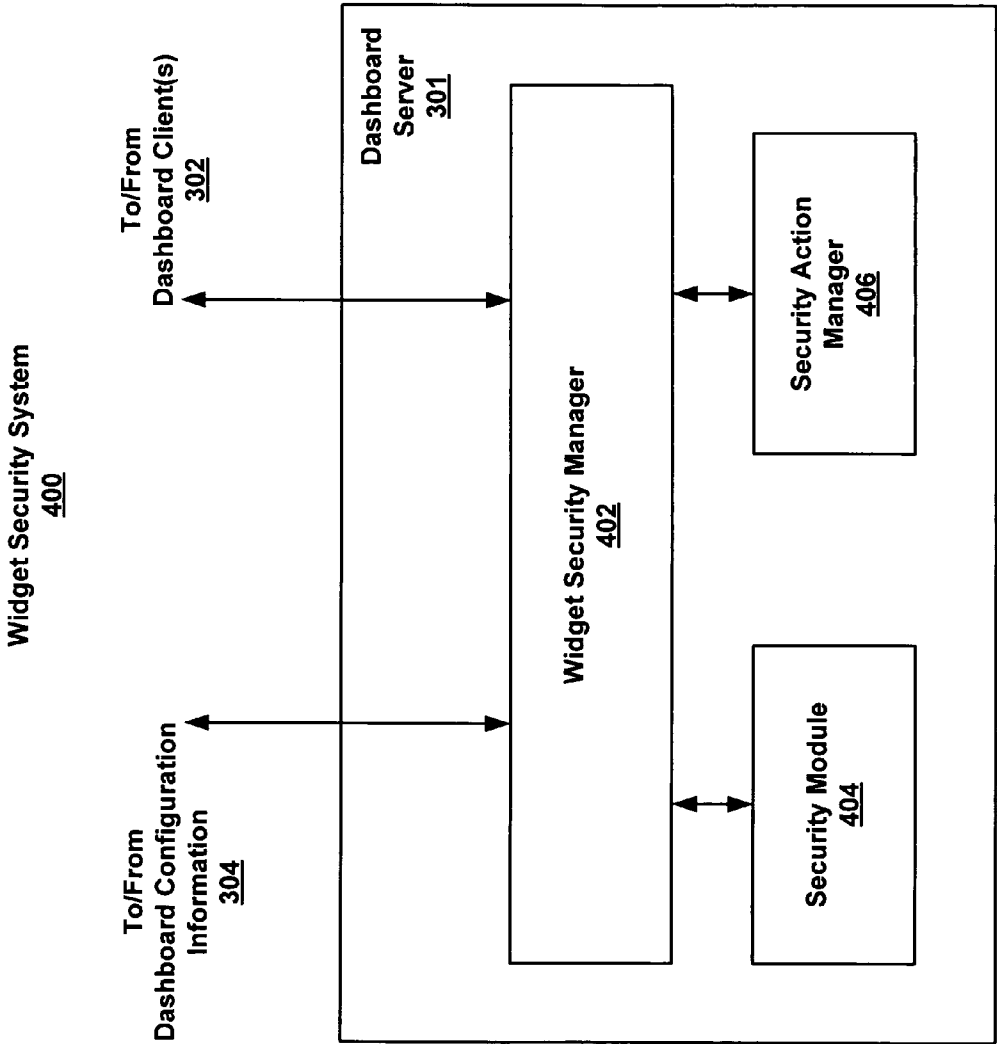


Fig. 4

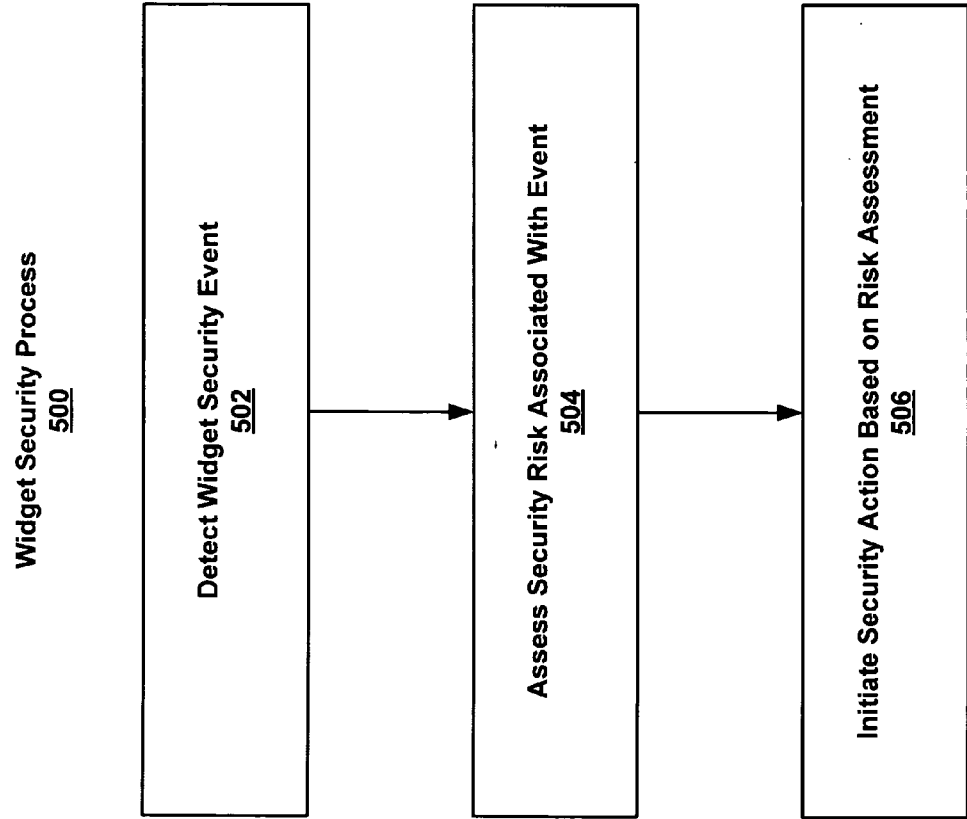


Fig. 5

WIDGET SECURITY**RELATED APPLICATIONS**

[0001] This application claims the benefit of priority from U.S. Provisional Patent Application No. 60/730,956, filed Oct. 27, 2005, for "Widget Security," which application is incorporated by reference herein in its entirety.

[0002] This application is related to the following U.S. patent applications, each of which is incorporated herein by reference in its entirety:

[0003] U.S. Provisional Patent Application No. 60/583,125, for "Procedurally Expressing Graphic Objects for Web Pages," filed Jun. 25, 2004;

[0004] U.S. patent application Ser. No. 10/874,829, for "User Interface for Assisting in the Installation of an Asset," filed Jun. 22, 2004;

[0005] U.S. patent application Ser. No. 10/877,968, for "Unified Interest Layer For User Interface," filed Jun. 25, 2004;

[0006] U.S. patent application Ser. No. 11/145,561, for "Application Clipper," filed Jun. 3, 2005;

[0007] U.S. patent application Ser. No. 11/145,560, for "Web View Applications," filed Jun. 3, 2005;

[0008] U.S. patent application Ser. No. 11/145,023, for "Clip View Applications," filed Jun. 3, 2005;

[0009] U.S. patent application Ser. No. 11/148,010, for "Preview and Installation of User Interface Elements in a Display Environment," filed Jun. 7, 2005;

[0010] U.S. Provisional Patent Application No. 60/734,016, for "Preview Including Theme Based Installation of User Interface Elements In A Display Environment," filed Nov. 4, 2005;

[0011] U.S. patent application Ser. No. 11/282,110, for "Preview Including Theme Based Installation of User Interface Elements In A Display Environment," filed Nov. 16, 2005;

[0012] U.S. Provisional Patent Application No. 60/737,899, for "Management of User Interface Elements In A Display Environment," filed Nov. 18, 2005; and

[0013] U.S. patent application Ser. No. 11/346,603, for "Multiple Dashboards," filed Feb. 1, 2006.

TECHNICAL FIELD

[0014] The disclosed implementations relate generally to graphical user interfaces.

BACKGROUND

[0015] A hallmark of modern graphical user interfaces is that they allow a large number of graphical objects or items to be displayed on a display screen at the same time. Leading personal computer operating systems, such as Apple MacOS X, provide user interfaces in which a number of windows can be displayed, overlapped, resized, moved, configured, and reformatted according to the needs of the user or application. Taskbars, menus, virtual buttons and other user

interface elements provide mechanisms for accessing and activating windows even when they are hidden behind other windows.

[0016] Although users appreciate interfaces that can present information on a screen via multiple windows, the result can be overwhelming. For example, users may find it difficult to navigate to a particular user interface element or to locate a desired element among a large number of onscreen elements. The problem is further compounded when user interfaces allow users to position elements in a desired arrangement, including overlapping, minimizing, maximizing, and the like. Although such flexibility may be useful to the user, it can result in a cluttered display screen. Having too many elements displayed on the screen can lead to "information overload," thus inhibiting the user to efficiently use the computer equipment.

[0017] Many of the deficiencies of conventional user interfaces can be reduced using "widgets." Generally, widgets are user interface elements that include information and one or more tools (e.g., applications) that let the user perform common tasks and provide fast access to information. Widgets can perform a variety of tasks, including without limitation, communicating with a remote server to provide information to the user (e.g., weather report), providing commonly needed functionality (e.g., a calculator), or acting as an information repository (e.g., a notebook). Examples of widgets can be found in U.S. patent application Ser. No. 10/877,968.

[0018] Due in part to their simplicity, hundreds of widgets have been developed and publicly distributed. Such ubiquitous use of widgets, however, has created a new challenge for system administrators, particularly in the realm of computer security. For example, since widgets behave like applications there is an opportunity for malicious activity. Malicious widgets can be configured by widget authors to access network resources, file systems and other vital system components to perform unauthorized activities. Such unauthorized activities can include accessing private information, logging user key strokes, redirecting the user to websites, installing a virus, deleting files, displaying inappropriate content and the like.

SUMMARY

[0019] In some implementations, a widget security method includes: detecting a security event associated with a widget; and initiating a security action based on the detected security event.

[0020] In some implementations, a widget security method detects a security event associated with a widget; assesses a risk associated with the detected security event; and initiates a security action based on the assessed risk.

[0021] In some implementations, a widget security method monitors widget activity; determines if the widget activity is consistent with an associated security policy; and responsive to the activity being consistent with an associated security policy initiates a security action.

[0022] In some implementations, a widget security method includes: initiating a download process for a widget; during the download process, determining a security event associated with the widget; and performing a security action based on the security event.

[0023] In some implementations, a widget security method includes: initiating an installation process for a widget; during the installation process, determining a security event associated with the widget; and performing a security action based on the security event.

[0024] Other implementations are disclosed relating to methods, systems, computer-readable mediums, apparatuses and devices.

BRIEF DESCRIPTION OF DRAWINGS

[0025] FIG. 1 is a block diagram of hardware architecture for a widget security system.

[0026] FIG. 2 is a flow diagram of a process flow for activating and using a dashboard.

[0027] FIG. 3 is a block diagram of software architecture for a widget security system.

[0028] FIG. 4 is a block diagram of a widget security system.

[0029] FIG. 5 is a flow diagram of widget security process.

DETAILED DESCRIPTION

Hardware Architecture Overview

[0030] FIG. 1 is a block diagram of an implementation of hardware architecture 100 for a widget security system. The architecture 100 includes a personal computer 102 coupled to a remote server 107 via a network connection 108 (e.g., local area network, wireless network, Internet, intranet, etc.). The computer 102 generally includes a processor 103, memory 105, one or more input devices 114 (e.g., keyboard, mouse, etc.) and one or more output devices 115 (e.g., a display device). A user interacts with the architecture 100 via the input and output devices 114, 115.

[0031] The computer 102 also includes a local storage device 106 and a graphics module 113 (e.g., graphics card) for storing information and generating graphical objects, respectively. The local storage device 106 can be a computer-readable medium. The term “computer-readable medium” refers to any medium that participates in providing instructions to a processor for execution, including without limitation, non-volatile media (e.g., optical or magnetic disks), volatile media (e.g., memory) and transmission media. Transmission media includes, without limitation, coaxial cables, copper wire, fiber optics, and computer buses. Transmission media can also take the form of acoustic, light or radio frequency waves.

[0032] While widget security is described herein with respect to a personal computer 102, it should be apparent that the disclosed implementations can be incorporated in, or integrated with, any electronic device that is capable of using widgets, including without limitation, portable and desktop computers, servers, electronics, media players, mobile phones, email devices, personal digital assistants (PDAs), network appliances, firewall applications and the like.

[0033] In some implementations, a widget security system can be provided as one or more plug-ins that are installed and run on the personal computer 102. The plug-ins are configured to interact with an operating system (e.g., MAC

OS X, WINDOWS XP, LINUX, etc.) and to perform the various widget security functions, as described with respect to FIGS. 4 and 5. A widget security system can also be implemented as one or more software applications running on the computer 102. In some implementations, a widget security system can be another widget that is configurable to communicate with other widgets, applications and/or operating systems. A widget security system can also be characterized as a security framework or model that can be implemented on various platforms and/or networks (e.g., client/server networks, stand-alone computers, portable electronic devices, mobile phones, etc.), and/or embedded or bundled with one or more software applications (e.g., email, media player, browser, etc.).

Dashboard Overview

[0034] FIG. 2 is a flow diagram of an implementation of a process for activating and using a dashboard. A dashboard layer (also referred to herein as a “unified interest layer” or “dashboard”) is used to manage and display widgets. A user can invoke a dashboard (202) by hitting a designated function key or key combination, or by clicking on an icon, or by selecting a command from an onscreen menu, or by moving an onscreen cursor to a designated corner of the screen. In response to such user input, the current state of the user interface is saved (203), the user interface is temporarily inactivated (204), an animation or effect is played or presented to introduce the dashboard (205) and the dashboard is displayed with one or more widgets (206). If applicable, a previous state of the dashboard is retrieved, so that the dashboard can be displayed in its previous configuration.

[0035] In some implementations, the dashboard is overlaid on an existing desktop user interface (UI). When the dashboard is activated, the existing UI may be faded, darkened, brightened, blurred, distorted, or otherwise altered to emphasize that it is temporarily inactivated. The existing desktop may or may not be visible behind the dashboard. The desktop can also be shrunk to a small portion of the display screen while the dashboard is active, and can be re-activated by clicking on it. In some implementations, the desktop is shrunk and presented as a widget. The desktop can be re-activated by clicking on the widget.

[0036] The user interacts with and/or configures widgets as desired (207). In some implementations, the user can move widgets around the screen, and can resize widgets if applicable. Some widgets are resizable and some have a fixed size. A widget author can specify whether a widget can be resized. Some widgets automatically resize themselves based on the amount or nature of the data being displayed. Widgets can overlap and or repel one another. For example, if the user attempts to move one widget to a screen position occupied by another widget, one of the widgets is automatically moved out of the way or repelled by the other widget.

[0037] The user dismisses the dashboard (208) by invoking a dismissal command, which causes the normal UI to return or re-present itself to the display screen. In some implementations, the dashboard is dismissed when the user presses a function key or key combination (which may be the same or different than the key or combination used to activate the dashboard), or clicks on a close box or other icon, or clicks on negative space within the dashboard (e.g., a space between widgets), or moves an onscreen cursor to a predefined corner of the screen.

[0038] In some implementations, the dashboard is automatically dismissed (i.e., without user input) after some predetermined period of time or in response to a trigger event. An animation or other effect is played or presented to provide a transition as the dashboard is dismissed (209). When the dashboard is dismissed, the current configuration or state of the widgets (e.g., position, size, etc.) is stored, so that it can be retrieved the next time the dashboard is activated. In some implementations, an animation or effect is played or presented when re-introducing the UI. The UI is restored to its previous state (210) so that the user can resume interaction with software applications and/or the computer operating system.

[0039] In some implementations, the dashboard is configurable. The user can select a number of widgets to be displayed, for example, by dragging the widgets from a configuration bar (or other user interface element) onto the dashboard. The configuration bar can include different types of widgets, and can be categorized and/or hierarchically organized. In some implementations, in response to the user dragging a widget onto the configuration bar, the widget is downloaded from a server and automatically installed (if not previously installed). In some implementations, certain widgets must be purchased, so the user is requested to provide a credit card number or some other form of payment before the widget is installed on the user's machine. In some implementations, widgets are already installed on the user's machine, but are only made visible when they have been dragged from the configuration bar onto the dashboard. The configuration bar is merely an example of one type of UI element for configuring the dashboard. Other configuration mechanisms can be used, such as an icon tray or menu system.

[0040] It should be apparent that there are many ways in which dashboards and widgets can be displayed other than those implementations described herein. For example, widgets can be displayed on any user interface or user interface element, including but not limited to desktops, browser or application windows, menu systems, trays, multi-touch sensitive displays and other widgets.

Multiple Dashboards

[0041] In some implementations, more than one dashboard is available. For example, the user can create and configure one dashboard to contain widgets related to work, and another for widgets related to personal matters. Different trigger events (e.g., different key combinations, menu selection, etc.) can be used for triggering the dashboards. State information for each dashboard can be saved enabling the dashboards to be restored to their previous respective configurations. Different dashboards can contain one or more of the same widgets. State information for a widget can be maintained separately for each dashboard in which the widget appears, or it can be commonly maintained across all dashboards in which the widget appears. Different dashboards can be available or "owned" for different users of the computer 102, such that each user can only access their own dashboard(s). A user can specify a dashboard as being available to other users, if desired. A user can also specify, for any or all of the dashboards he or she creates, whether other users are permitted to make changes to the dashboard(s).

Software Architecture Overview

[0042] FIG. 3 is a block diagram of an implementation of software architecture 300 for a widget security system. The software architecture 300 generally includes a dashboard server 301 one or more dashboard clients 302, and one or more widgets 303. The server 301 and/or clients 302 use dashboard configuration information 304 to specify configuration options for displaying the widgets 303 including access levels and the like (if applicable). Such configuration information can include information for two or more dashboards configured by the same user or by different users.

[0043] In some implementations, the widgets 303 are displayed using HTML and related web technology. The dashboard server 301 manages and launches the dashboard client 302 processes. Each dashboard client 302 loads a widget 303 (e.g., an HTML webpage) and related resources needed to display the page. In some implementations, the dashboard clients 302 display the widgets 303 without a conventional window frame, menu bar, or other components typically associated with on-screen windows. This technique provides a clean, straightforward display of the overall dashboard to reduce confusion and clutter. The dashboard clients 302 display their respective widgets 303 by rendering web pages into a "WebView," as described in U.S. patent application Ser. No. 11/148,010. The size of each WebView is defined as metadata associated with the corresponding widget 303. The server 301 provides data for rendering a separate layer that can be overlaid on the normal desktop of the user interface. The widgets 303 are rendered into the separate layer which is drawn on top of the normal desktop, so as to partially or completely obscure the desktop while the dashboard is active.

Dashboard Server

[0044] The dashboard server 301 can be a stand-alone process or embedded in another process. The server 301 can be located at the computer 102 or at the remote server 107. In some implementations, the server 301 provides functionality for one or more processes, including but not limited to: non-widget UI management, window management, fast login, event management, loading widgets, widget arbitration, Core Image integration and widget preference management, as described in U.S. patent application Ser. No. 11/148,010.

Dashboard Client

[0045] In some implementations, a dashboard client 302 is a process that uses, for example, objects that are defined as part of a development environment, such as Apple Computer's Cocoa Application Framework (also referred to as the Application Kit, or AppKit) for the MacOS X operating system. In some implementations, the dashboard clients 302 can be implemented as simplified browser screens that omit conventional interface features such as a menu bar, window frame, and the like.

Widget Format

[0046] In one implementation, each widget 303 is implemented as an HTML file. The HTML file can reference other local and remote resources such as style sheets (e.g., Cascading Style Sheets), other HTML files, JavaScript files, images, and the like. Widgets 303 can be implemented

using, for example, a flat bundle file format or a packaged HTML file format. In some implementations, the Flat Bundle format includes an info.plist file.

[0047] The Info.plist file describes a widget 303 and provides an identifier for a widget 303. Table I provides an example of Info.plist file contents.

TABLE I

Example of Info.plist File Contents		
Key	Type	Description/ Value
CFBundleIdentifier	CFString	com.apple.widget <widget name>
CFBundleName	CFString	Name of the widget.
MainHTML	CFString	Name of main HTML resource.
Width	CFNumber	Default width of the widget.
Height	CFNumber	Default height of the widget.
DefaultImage	CFString	Resource name of default PNG file.
Plugin (optional)	CFString	Resource name of native plug-in.
AllowFileAccessOutsideofWidget	Boolean	Access to files across the file system; limited by the users permissions.
AllowFullAccess	Boolean	Access to the file system, Web Kit and standard browser plug-ins, Java applets, network resources, and command-line utilities.
AllowInternetPlugins	Boolean	Access to Web Kit and standard browser plug-ins.
AllowJava	Boolean	Access to Java applets.
AllowNetworkAccess	Boolean	Access to any resources that are not file based.
AllowSystem	Boolean	Access to command- line utilities using widget script object.

[0048] The keys AllowFileAccessOutsideofWidget, AllowFullAccess, AllowInternetPlugins, Allowjava, AllowNetworkAccess, and AllowSystem are Boolean types that can be set by a widget author to enable certain levels of resource access.

Widget Security

[0049] FIG. 4 is a block diagram of an implementation of a widget security system 400. The widget security system 400 can be part of the dashboard server 301 (as shown in FIG. 4) and/or part of one or more dashboard clients 302. Alternatively, the widget security system 400 can be part of the operating system 305, or a separate and independent process or application (e.g., a plug-in or another widget). The various components of the widget security system 400 can be implemented on computing devices, non-computing devices and other electronic devices, including but not

limited to a single personal computer, multiple computers via one or more network connections, and portable devices (e.g., media players, mobile phones, email devices, PDAs, embedded devices, etc.).

[0050] The widget security system 400 generally includes a widget security manager 402, a security module 404 and a security action manager 406. In some implementations, the widget security manager 402, the security module 404 and the security action manager are code or instructions stored on a computer-readable medium and executed by one or more processors. Some or all of the code can be part of an operating system, application, plug-in, driver, object, etc. In some implementations, the widget security manager 402, the security module 404 and the security action manager 406 are processes running in a client/server framework. For example, the security manager 402 can be a server process and the security module 404 and the security action manager 406 can be client processes that receive security events and other information served by the widget security manager 402 server process.

Widget Security Manager

[0051] The widget security manager 402 monitors widgets 303 for security events. The widget security manager 402 can be automatically invoked by a dashboard process or operating system as a background process. It can also be a widget that is invoked when, for example, it is dragged from a configuration bar into a dashboard.

[0052] A security event can occur during download, installation and post-launch. Security events generally include any events that could potentially compromise the security of the host system and/or other computers and devices coupled to the host system. Examples of security events include attempts or requests to access restricted resources, such as file systems, network resources, WebKit, browser plug-ins, media players, command-line utilities, Java applets, etc. Other events include but are not limited to: 1) downloading, installing or launching unauthorized widgets, 2) attempting to review unauthorized content generated or presented by widgets, 3) attempting to direct or redirect a user to a website, 4) attempting to take precedence over standard widgets (e.g., a widget disguising itself as a another widget), 5) activating a timed attack in response to an elapsed timer or user input, 6) logging user keystrokes, 7) harvesting personal information, and 8) initiation of unauthorized transactions.

[0053] In some implementations, the widget security manager 402 is responsible for handling multiple security events from multiple widgets and dashboards, and includes appropriate data structures for storing events and related information. In some implementations, the widget security manager 402 creates a security event queue (not shown) for each widget that is installed to store and process multiple security events in an organized manner. The widget security manager 402 can also include components for recording security events (e.g., security event logs) for further analysis by, for example, a system administrator. In some implementations, security event information and associated data structures are saved to local storage 106. The security event information can be part of the dashboard configuration information 304 or it can be stored separately in one or more data structures on the computer 102. The security event information can

also be stored on a network server to facilitate global security management for networked computer systems.

Security Module

[0054] The security module **404** includes components for handling various aspects of widget security risk assessment, and is generally responsible for enforcing a security model or policy. In some implementations, there are at least three contexts where security events can occur: administration, content and resource access. Each of these contexts are described in more detail with respect to FIG. 5.

Security Action Manager

[0055] The security action manager **406** selects and initiates appropriate security actions based on input from the security module **404**. Generally, security actions address security risks, as defined by a security module or policy. Examples of security actions include but are not limited to: 1) sending a warning message or alert to the user regarding a potential violation of a security policy, 2) requesting a password or personal identification from the user before allowing an event to proceed, 3) disabling the user's dashboard or widget, or 4) changing the properties of the widget (e.g., mark the widget as not trusted).

Widget Security Process Flow

[0056] FIG. 5 is a flow diagram of widget security process **500**. The widget security system **400** provides security for various activities relating to widgets and dashboards, including downloading, installation, previewing, launching and other events.

[0057] In some implementations, the process **500** is a background process that runs continuously when the dashboard is running and/or widgets are active. The process **500** can be turned off or idled when no widgets are running to save processor cycles. The process **500** is not limited to platforms that run dashboards. Rather, the process **500** can also run on a variety of electronic devices (e.g., media players, mobile phones, firewall appliances, embedded systems, etc.) and can integrate and/or communicate with a variety of operating systems and applications, including operating systems and applications that do not provide dashboard or dashboard-type functionality.

[0058] The steps of process **500** do not have to occur in any particular order and can run concurrently as multiple processes in multithreading environments on one or more processors, depending upon the computing platform.

Detection of Security Events

[0059] The process **500** begins each time a widget security event is detected (**502**). In some implementations, security events are detected by the widget security manager **402** which communicates with the operating system **305** and/or dashboard clients **302**. For example, whenever a widget is downloaded, installed or launched, the operating system **305** informs the widget security manager **400** of the event. If the event occurs after a widget is launched, then the dashboard clients **302** can also provide the widget security manager with security event information. Thus, dashboard clients **302** are particularly useful for guarding against rogue widgets that engage in malicious activities after launch.

Download Security Events

[0060] It is often desirable to detect security events early. In some implementations, the widget security manager **402** monitors the widget download process for security events. For example, the widget security manager **402** can examine files (e.g., info.plist) associated with a widget download to determine if the widget is on a "black list" of malicious widgets. If the widget is included on the list, then the security action manager **406** can instruct the operating system to terminate the download process and/or provide a suitable warning to the user. Alternatively, the widget security manager **402** can be configured to allow widget downloads only from authorized or trusted websites. For example, a system administrator may set preferences that enable users to download widgets only from certain designated entities or individuals.

[0061] In some implementations, widgets can be downloaded as a bundle of files that are signed by a third party Certification Authority (CA) using known digital certificate techniques. The digital certificate can be used to identify trusted web server for downloading widgets over networks, such as the Internet. For example, before a user can download a widget, the web security manager **402** can request a secure session (SSL). The widget web server responds by sending the widget security manager **402** its server certificate (which includes a public key). The widget security manager **402** verifies that the web server's certificate is valid and has been signed by a CA (e.g., Entrust), whose certificate is stored at the computer **102**. It also verifies that the CA certificate has not expired. If the certificate is valid, the widget security manager **402** generates a one-time, unique "session" key and encrypts it with the web server's public key. The widget security manager **402** (or a browser) sends the encrypted session key to the web server so that it will have a copy. The web server encrypts the message using its private key and recovers the session key. Once this SSL "handshake" is complete, a secure communication channel is established for downloading widgets.

[0062] In some implementations, the widget security manager **402** requests a manifest from a trusted website for downloading widgets. After successful completion of an SSL handshake, the widget security manager **402** downloads a manifest describing locations where the widgets can be downloaded (e.g., a URL to a download page). The widgets to be downloaded can be signed by a CA to detect a "man in the middle" attack. In some implementations, a hash function is used to generate and verify hashes (e.g., SHA-1) for widget bundles to verify the integrity of widget files after download.

[0063] In some scenarios, a malicious widget may attempt to download suspicious files into folders not normally associated with widgets. In such a case, the operating system can detect such activities and notify the security action manager **406**. The security action manager **406** can then take appropriate action, such as deleting or quarantining the suspect files and/or alerting the user of the security event.

[0064] In some implementations, dashboards communicate with a trusted network server (e.g., a server on the Internet or an intranet) to gain access to a "black list" of widgets stored on the server that are known security risks. Since the "black list" is stored on the network server it can be accessed by multiple user systems or devices without the

user systems or devices downloading a copy of the list each time the list is updated. If a widget is on the “black list,” then the security action manager **406** can initiate the appropriate action (e.g., disable or deactivate the widget, prevent installation, etc.). Alternatively, the “black list” or a portion of the “black list” can be stored at the user system or device.

[0065] In some implementations, user systems or devices are automatically alerted of the “black list,” or updates to the “black list,” using known software update technology which notifies user systems or devices when software updates are available (e.g., Apple’s “Software Update” application which ships with its Mac OS X operating system). For example, upon receipt of an alert or other notification, a network connection can be established to provide dashboards (e.g., the widget security manager **402**) access to the “black list.” Using information provided in the “black list,” the security action manager **406** can initiate an appropriate security action (e.g., disable or deactivate the widget, terminate or prevent installation or launch, etc.). If the user has suitable privileges, the user can disable or deactivate the automatic alert/notification function using a preference pane or other input mechanism. A user can also specify various properties of the automatic alert/notification, such as whether to display a message or alert the user before establishing a connection with a network server to access the “black list.”

Installation Security Events

[0066] In some implementations, security events are detected during widget installation. These security events can include (but are not limited to) detecting a potential to access to host or network resources, installation of a virus, spyware, adware or other undesirable applications, and displaying offensive content. If the widget security manager **402** detects such events, it instructs the security action manager **406** to take appropriate security actions. For example, anti-virus, anti-spyware, and anti-adware applications can be invoked by the security action manager **406** as needed to seek out and remove suspect applications and files. Content filters can also be invoked to detect offensive content (e.g., adult content), as described with respect to FIG. 5.

[0067] In some implementations, the widget security manager **402** invokes a security protocol in response to an installation security event which can include storing various state or status information about the widgets for use in subsequent security assessments and actions. For example, the widget security manager **402** can store data related to the state of the widget at installation, including a list of resources that the widget is allowed to access, locations of widget files, security risk levels and the like. After the widget is launched, the installation state can be compared to future widget states to determine if the widget is functioning differently than expected when first launched. For example, prior to launch a widget may have requested access to network resources but not the host file system, i.e., the widget’s installation state. If after launch the widget attempts to access the host file system, i.e., a current widget state, the current widget state is compared with the widget’s installation state, and the result of the comparison will be used to determine if a security event exists.

Post-Launch Security Events

[0068] Even if no security events are detected during download or installation, widgets can be continuously moni-

tored by the widget security manager **402** after launch. For example, a widget may contain a virus that is launched at some predetermined time after launch or in response to user input (e.g., a particular key combination) and attempts to access host or network resources. In such a scenario, the operating system **305** informs the widget security manager **402** of the access attempt, the security module **404** determines the level of risk for the event, and the security action manager **406** initiates the appropriate security action based on the risk assessment (e.g., disable the widget).

[0069] In multiple dashboard implementations, certain widgets may be restricted from being installed in certain dashboards. Likewise, users may be restricted from launching certain widgets in dashboards for other users. For example, a guest user on a computer may be restricted from launching certain widgets in the owner’s personal dashboards. In such a scenario, the widget security manager **402** is informed of the guest login by the operating system **305**. The widget security manager **402** instructs the security action manager **406** of the attempted widget installation by a guest user, and the security action manager **406** takes the appropriate security action (e.g., disable the widget when it is in the owner’s personal dashboard).

Risk Assessment

[0070] In response to a detected security event, the security module **404** assesses the security risk associated with the security event (**504**). In some implementations, risk assessment includes identifying risk based on characteristics of the security event and one or more policies that are associated with the type of event, then determining whether a breach of security has occurred or would occur if the event proceeds. For example, if a launched widget attempts to access a host or network resource, and such access is unauthorized, then the security module **404** informs the security action manager **406** of the attempted access, so the appropriate security action can be initiated (e.g., disable the widget).

[0071] In some implementations, risk assessment includes determining whether a widget’s request to access a network resource is reasonable given the widget’s class or type. For example, it may be unreasonable for a “clock” widget to request access to network resources. Likewise, it may be reasonable for a “browser” widget to request access to standard browser plug-ins but unreasonable to request access to command-line utilities. Examples of widget classes include: games, date and time, news feeds (e.g., RSS feeds), system utilities (e.g., a password maker, battery level indicator, wireless connection indicator), sight and sound (e.g., media players), cam viewers, widget tools (e.g., calculator, currency converter), application enhancers (e.g., status indicators), search tools (e.g., browser, dictionary), etc.

[0072] In some implementations, a widget can be classified based on information bundled with the widget (e.g., included in the info.plist). If the widget author has requested network access (e.g., AllowNetworkAccess=TRUE), then the widget security manager **400** would classify the widget accordingly.

[0073] In some implementations, a widget could be classified by the user during a preview session. Preview sessions are described in U.S. patent application Ser. No. 11/148,010. Since information provided by the widget author may inten-

tionally or unintentionally not match the advertised functions of the widget, the user can be prompted to provide feedback during a preview session that could be used to classify the widget. For example, the user could be requested to categorize the widget class through a dialogue box or equivalent UI element for receiving user input.

[0074] Still another opportunity to classify a widget could occur by running the widget through a widget clearing house prior to installation. A widget clearing house can be, for example, a trusted website that maintains a database of widget information for safe and/or malicious widgets. For example, the security module 404 could send a request to a clearing house (e.g., via an HTTP request) which could provide an accurate description of the widgets functions that can be used to classify the widget.

[0075] In some implementations, the security module 404 examines the application code bundled with the widget to determine its class. For example, if the widget bundle or package contents includes JavaScript (indicated by a .js extension), then the code is examined for certain calls that invoke resources, such as widget.openApplication, widget.system, or widget.openURL.

[0076] In some implementations, the security module 404 can determine whether a widget is included on a “black list” of malicious widgets that are known to engage in malicious activity, or alternatively, on a “white list” of safe widgets. If the widget is on a list, then the security module 404 uses such information to assign an appropriate risk level, and instruct the security action manager 406 to initiate an appropriate security action, if any. The lists can be stored locally (e.g., local storage device 106) or retrieved or accessed from a remote server (e.g., remote server 107).

[0077] In some implementations, risk assessment is a dynamic process. The security module 404 may change the security risk associated with a widget based on new activities. For example, a widget can be declared safe during installation, but later release a virus after installation. In this scenario, the widget security manager 402 detects the suspect activity as a security event and the security module 404 re-assesses the risk level of the widget based on the security event. The risk re-assessment can include comparing the widget’s installation state with the widget’s current state and re-assessing the risk based on the result of the comparison. Historical information (e.g., security event log) can also be used re-assess risk by determining patterns of activity associated with certain levels of risk.

Security Actions

[0078] After the security event has been assessed by the security module 404, one or more security actions are initiated and managed by the security action manager 406 based on the risk assessment (506). In some implementations, the security action manager 406 accesses an indexed list of security actions (not shown) which are associated with certain levels and/or types of risk assigned by the security module 404.

[0079] For example, if the security event is classified as “low risk” then the security action may be an alert message sent to the user, or a request for a password, depending upon the security event and/or widget class. If the security event is classified as “high risk” then the security action could include disabling the widget and/or dashboard.

[0080] In some implementations, performing a security action includes informing a user about the security risk; requesting a response from the user; selecting a security action based on the response; and initiating the selected security action. For example, during installation the security module 404 detects a potential security risk due to access requests made by a widget (e.g., AllowFullAccess=TRUE), then the security action manager 406 issues a warning message to the user. One example of a suitable warning message can be: “This widget is requesting full access to all host system and network resources. Would you like to continue with the installation or quit?” Depending upon the user’s response, the security action manager 406 either allows the installation to continue or terminates the installation process.

[0081] In some implementations, the security action manager 402 allows a time-based trial period in which a widget has limited access to host or network resources. For example, a user may be allowed to use a suspect widget for a predetermined amount of time (e.g., 24 hours) during which time the widget security manager 402 monitors the widget’s activity. If no security events are detected, then widget is allowed full or additional access to host and/or network resources.

[0082] It should be apparent that there are other situations where a warning message would not be appropriate for the risk level of the security event. For example, if the resource request made by the author is completely inconsistent with the widget class, then it may be appropriate to automatically terminate the installation without the user’s permission. For example, if a “game” widget is attempting to access the host computer’s file system, then the security action manager 406 would terminate the installation since such a request would be inconsistent with the resource needs of a “game” widget. In such a scenario, the security action manager 406 could send a message to the user explaining why the widget was not installed. In some implementations, the security action manager 406 logs the security event and sends the log to a clearing house as a possible malicious widget. The security action manager 406 adds the widget to a local “black list” of malicious widgets for use in future security risk assessment and actions.

[0083] As previously described, the security module 404 is responsible for assessing various risks. These security risks occur in a variety of contexts. Three of these contexts (administrative, content and resource access) are described, in turn, below.

Administrative Security

[0084] Administrative security includes security events related to user privileges for downloading, installing and launching. This type of security would typically be found in networked computer systems that are managed by a system administrator, such as corporate computer networks that have multiple users sharing network resources (e.g. client/server topology). Administrative security is also applicable to stand-alone computers that are shared by multiple users.

[0085] In a networked environment, it may be desirable to set and store privileges with respect to widgets and dashboards. For example, a system administrator can use the widget security system 400 to enforce a security policy by setting user privileges via an administrative UI (not shown).

In such an implementation, the widget security system **400** can be installed on a server coupled to a centralized system administration computer. Such a configuration could include a central repository of widget and dashboard information, as well as provide centralized control for management and enforcement of widget security policies. An example administrative security policy includes, for example, restricting certain users or user groups from downloading, installing, launching, and/or using certain widgets based on different security criteria. Examples of administrative security criteria include but are not limited to: the widget's class, the location of the host machine, a profile of the user or user group, and any other suitable criteria for assigning privileges.

[0086] For example, if a user attempts to download, install and/or launch a widget, the widget security manager **402** detects an administrative security event. The security module **404** receives notification of the event from the widget security manager **402** and determines if the user has the appropriate privileges to proceed with the event. If the user has the appropriate privileges, then the event can proceed. Otherwise, the security module **404** informs the security action manager **406** of the event and the level of security risk, so that the security action manager **406** can initiate one or more security actions.

[0087] Different privileges can be granted to different users or user groups as desired, and each user or user group can have multiple sets of privileges that can be invoked based on a variety of factors, as described below.

[0088] Privileges can be granted or denied based on widget class. In some implementations, widgets are tagged so as to identify them with a particular class of widgets (e.g., games, utilities, viewers). For example, a game widget may be restricted from being installed on certain host machines (e.g., an employee computer) or from being used during certain times (e.g., during working hours). A time-based privilege may specify, for example, on/off times or allocate an amount of time to use a widget. Time-based privileges can also be used to control bandwidth and access to limited resources. For example, if hundreds of users are concurrently using a widget that accesses a limited resource (e.g., a single server), then by limiting use of the widget to certain groups of users during certain times of the day (e.g., through privilege settings) bandwidth and access problems can be reduced.

[0089] Privileges can be granted or denied based on the location of the host machine. For example, if the host machine is located in a secure facility, then some or all network access privileges can be denied or restricted. In some implementations, the location of a host machine (e.g., a lap top computer) can be tracked using a position locator (e.g., global positioning system (GPS), network-based positioning, etc.), and the user's privileges can be changed based on the location of the host machine.

[0090] Privileges can be granted or denied based on the context or environment in which the widget will run. For example, if a user is using a word processor application, then the user's privileges would include the ability to launch and use widgets that enhance word processing applications (e.g., dictionary or encyclopedia widgets), but restrict widgets that are outside the current context from being launched or used.

[0091] In multi-dashboard environments, each dashboard could be associated with a different set of privileges. One

dashboard could be for work-related widgets (e.g., calculator, dictionary) and another dashboard could contain widgets for personal use (e.g., media player). In such environments, it may be advantageous to have different sets of privileges for each dashboard. This could be useful for shared computers where a guest login would have access to work-related dashboards but not the owner's personal dashboards. Multiple dashboard functionality is described in U.S. patent application Ser. No. 11/148,010.

[0092] It should be apparent that any number of factors can be used to set privileges related to widgets and dashboards based on security policies, user preferences and/or user profiles.

Content Security

[0093] Content security is concerned with controlling access to content provided by widgets (e.g., media players, browsers). For example, an employer (or parent) may want to control access to certain content. In some implementations, the security module **404** receives content security events from, for example, a content filter. The content filter can be part of the widget security system **400** or external to the widget security system **400**. A content filter detects certain content (e.g., adult content, personal health care information) using a variety of known techniques (e.g., URL or keyword filtering, dynamic content filter, etc.). In some implementations, the content filter is configured to send reports to the security module **404** via the widget security manager **402**. Alternatively, the content filter can report directly to the security module **404**. In some implementations, the content filter sends raw data to the security module **404**, which then uses the raw data to make a risk assessment. The security action manager **406** initiates and manages one or more security actions (e.g., disable the widget) based on the risk assessment, reports and/or user profiles and content filter information.

[0094] In business environments, it may be desirable to restrict the class of users who can access certain content and how that content is viewed, or limit the amount or type of content that can be accessed. For example, content providers may want to limit a user's access to certain content. Such content can be distributed using a digital rights management (DRM) framework. The security module **404** enables secure delivery of protected content for playback through a widget by providing a secure environment that can enforce DRM rules (e.g., via the security action manager **406**).

[0095] In some implementations, the DRM rules associated with the content can be added to the security policy enforced by the security module **404**. For example, if the content's DRM rules do not allow copying, and a widget requests access to command-line copy utilities, then the security action manager **406** will be alerted to the violation, so it can take appropriate security actions (e.g., disable widget).

Resource Access Security

[0096] Widgets can request access to various resources (e.g., file systems, networks, command-line utilities, etc.). Resources can be restricted based on the widget's class, as previously described with respect to administrative security. It may be desirable, however, to partially restrict access to

resources based on whether the widget is being previewed, running in a trial-period or in full operation.

[0097] In some implementations, if a widget is being previewed the widget security manager 402 automatically disables access to certain resources that are required for previewing widget functionality prior to installation. For example, a widget's request to access a file system or a command-line utility can be denied without diminishing the user's preview experience. During normal widget operation, additional access to resources can be permitted as requested.

[0098] Various implementations of a widget security system have been described with respect to FIGS. 1-5. It should be apparent that these implementations can be modified and still remain within the scope of the appended claims.

What is claimed is:

1. A widget security method, comprising:
 - detecting a security event associated with a widget;
 - assessing risk associated with the detected security event; and
 - initiating a security action based on the assessed risk.
2. The method of claim 1, wherein detecting includes detecting unauthorized access to resources.
3. The method of claim 1, wherein detecting includes detecting unauthorized access to content.
4. The method of claim 1, wherein assessing the risk includes:
 - identifying risks based on characteristics of the event and one or more policies associated with the event; and
 - identifying a risk based on the results of the comparison.
5. The method of claim 1, wherein assessing the risk is selected from the group including reviewing information associated with the widget, comparing the widget information with a user profile, and examining programming code associated with the widget.
6. The method of claim 1, wherein assessing the risk includes:
 - monitoring widget activity; and
 - identifying a risk based on the activity.
7. The method of claim 6, wherein monitoring widget activity includes logging widget activity.
8. The method of claim 1, wherein assessing the risk includes:
 - retrieving a list of widgets;
 - determining if the widget is on the list; and
 - assessing risk associated with the widget based on whether the widget is included on the list.
9. The method of claim 1, wherein performing a security action includes:
 - informing a user about the security risk;
 - requesting a response from the user;
 - selecting a security action based on the response; and
 - initiating the selected security action.
10. The method of claim 1, wherein performing a security action is selected from the group including disabling the widget, notifying the user of the security event, and changing resource access rights associated with the widget.

11. A widget security method, comprising:
 - monitoring widget activity;
 - determining if the widget activity violates a security policy; and
 - responsive to the activity violating a security policy, initiating a security action.
12. A widget security method, comprising:
 - monitoring widget activity;
 - determining if the widget activity is consistent with an associated security policy; and
 - responsive to the activity being consistent with an associated security policy, initiating a security action.
13. A widget security method, comprising:
 - receiving notification of a security event in response to user interaction with a widget; and
 - initiating a security action based on the security event and one or more user privileges.
14. A computer-readable medium having instructions stored thereon, which, when executed by a processor in a widget security system, causes the processor to perform the operations of:
 - detecting a security event associated with a widget;
 - assessing risk associated with the detected security event; and
 - performing a security action based on the assessed risk.
15. The computer-readable medium of claim 14, wherein detecting includes detecting unauthorized access to resources.
16. The computer-readable medium of claim 14, wherein detecting includes detecting unauthorized access to content.
17. The computer-readable medium of claim 14, wherein assessing the risk includes:
 - identifying risks based on characteristics of the event and one or more policies associated with the event; and
 - identifying a risk based on the results of the comparison.
18. The computer-readable medium of claim 14, wherein assessing the risk is selected from the group including reviewing information associated with the widget, comparing the widget information with a user profile, and examining programming code associated with the widget.
19. The computer-readable medium of claim 14, wherein assessing the risk includes:
 - monitoring widget activity; and
 - identifying a risk based on the activity.
20. The computer-readable medium of claim 19, wherein monitoring widget activity includes logging widget activity.
21. The computer-readable medium of claim 14, wherein assessing the risk includes:
 - retrieving a list of widgets;
 - determining if the widget is on the list; and
 - assessing risk associated with the widget based on whether the widget is included on the list.
22. The computer-readable medium of claim 14, wherein performing a security action includes:

informing a user about the security risk;
requesting a response from the user;
selecting a security action based on the response; and
initiating the selected security action.

23. The computer-readable medium of claim 14, wherein performing a security action is selected from the group including disabling the widget, notifying the user of the security event, and changing resource access rights associated with the widget.

24. A widget security system, comprising:

a widget security manager configured for detecting a security event associated with a widget;

a security module coupled to the widget security manager and configured for assessing risk associated with the detected security event; and

a security action manager coupled to the widget security manager and configured for initiating a security action based on the assessed risk.

25. A widget security method, comprising:

detecting a security event associated with a widget; and
initiating a security action based on the detected security event.

26. A computer-readable medium having instructions stored thereon, which, when executed by a processor in a widget security system, causes the processor to perform the operations of:

detecting a security event associated with a widget; and
initiating a security action based on the detected security event.

27. A widget security method, comprising:

initiating a download process for a widget;

during the download process, determining a security event associated with the widget; and

performing a security action based on the security event.

28. A computer-readable medium having instructions stored thereon, which, when executed by a processor in a widget security system, causes the processor to perform the operations of:

initiating a download process for a widget;

during the download process, determining a security event associated with the widget; and

performing a security action based on the security event.

29. A widget security method, comprising:

initiating an installation process for a widget;

during the installation process, determining a security event associated with the widget; and

performing a security action based on the security event.

30. A computer-readable medium having instructions stored thereon, which, when executed by a processor in a widget security system, causes the processor to perform the operations of:

initiating an installation process for a widget;

during the installation process, determining a security event associated with the widget; and

performing a security action based on the security event.

* * * * *